

-Rapport sur eLearning siteweb-

Introduction :

In this project, I reinvented everything we learned in class like:

- ✓ Use of tables, lists and forms;
- ✓ JavaScript.
- ✓ Use of JavaScript events to detect user actions, and to manipulate HTML elements using DOM functions.
- ✓ JSON; cookies...

My objective :

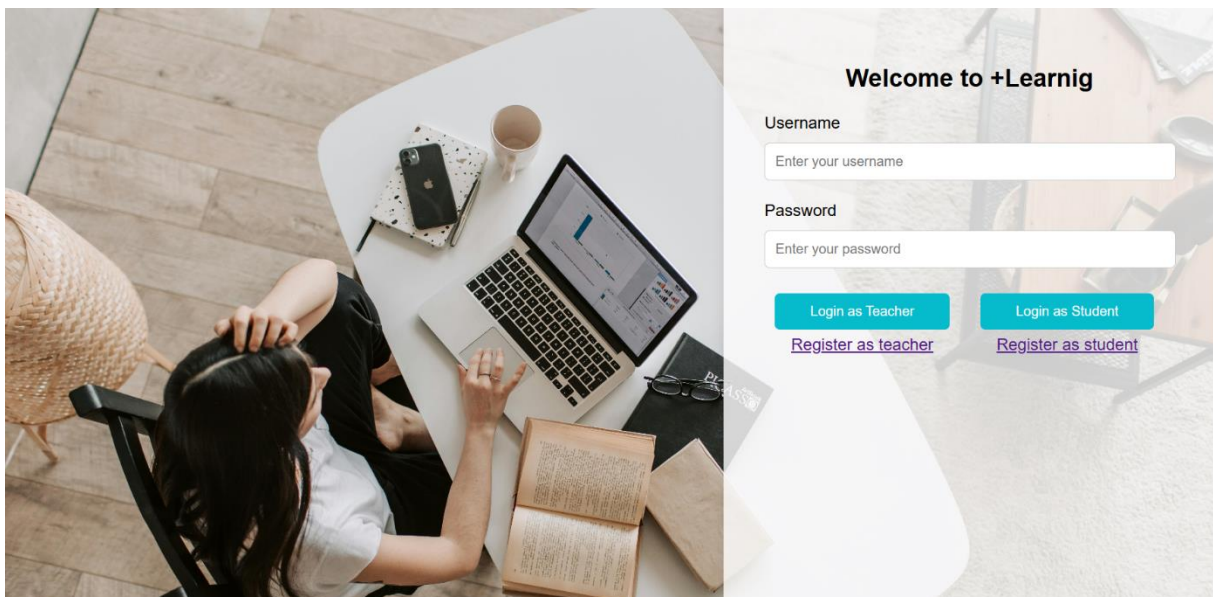
Welcome to eLearning place. The global destination for a better and an exciting learning.

By signing up students, my users will get all of our interesting courses and will start studying at just one click from home which the main reason why I created my website to facilitate the learning process for them.

On the other side my website cannot be that successful without teachers that will provide courses and have access to the list of the subscribed students as well as their payment methods.

All the work I have done was to assist both teachers and students to communicate and help the teacher to keep track of each subscribed student.

Input interface:



➤ Le code js du login:

```
var form = document.getElementById("LoginForm");

var teacherSubmit = document.getElementById("teacherSubmit");
var studentSubmit = document.getElementById("studentSubmit");

// when the teacher submit button is clicked we look in local storage for the teacher that has the name and password entered by the use
teacherSubmit.addEventListener("click", function(event) {
    event.preventDefault();
    var username = document.getElementById("username").value;
    var password = document.getElementById("password").value;
    const teachers = JSON.parse(localStorage.getItem('teachers')) || [];
    var teacher = teachers.find(teacher => teacher.username === username && teacher.password === password);

    if (teacher !== null) {
        document.getElementById("Message").textContent = "Teacher login successful!";

        // if we find the teacher we store his informations in session storage
        sessionStorage.setItem('teacher', username);
        sessionStorage.setItem('IsStudent', false);
        sessionStorage.setItem('firstName', teacher.firstName);
        sessionStorage.setItem('middleName', teacher.middleName);
        sessionStorage.setItem('username', teacher.username);
        sessionStorage.setItem('password', teacher.password);
        sessionStorage.setItem('Level', teacher.subject);
    }
});
```

Login JavaScript code handle user authentication for a login form on my webpage. This code provides a basic authentication mechanism for my webpage, allowing teachers and students to log in with their credentials and directing them to their respective pages upon successful authentication.

➤ Here's what it does:

➤ Initialization:

When the DOM content is fully loaded, it sets up some initial session storage items with default values. These items represent user information such as whether the user is a student or a teacher, their names, usernames, passwords, and levels.

➤ Event Listeners:

It sets up event listeners for the submit buttons (teacherSubmit and studentSubmit) of the login form.

When the teacher submit button is clicked, it retrieves the entered username and password, then checks if there's a matching teacher entry in the local storage.

If a matching teacher is found , it updates session storage with the teacher's information and redirects the user to a teacher-specific page (teacherpage.html).

If no matching teacher is found, it displays an error message.

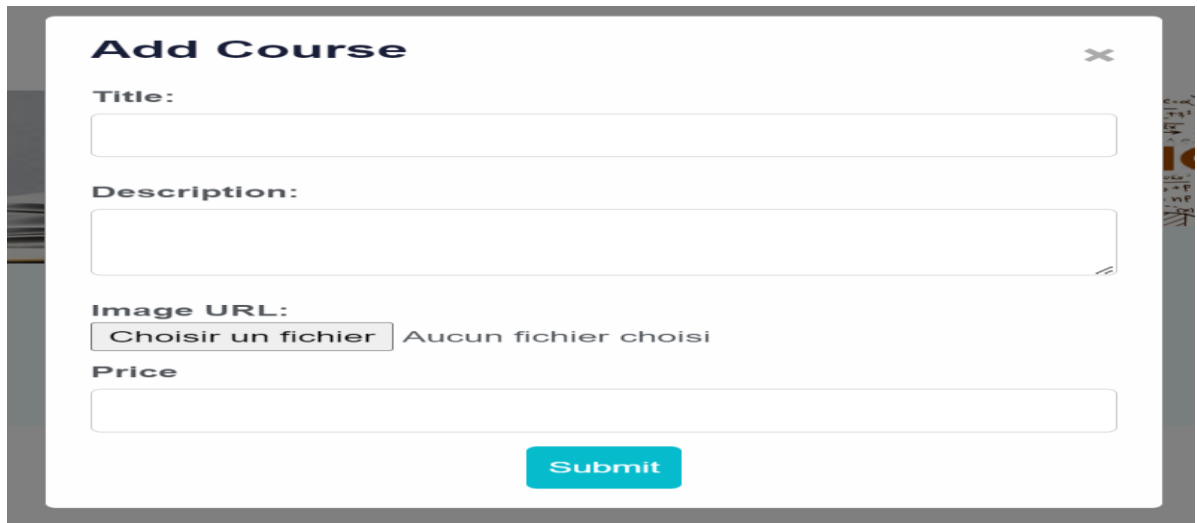
Similarly, when the student submit button is clicked, it follows the same process for student authentication and redirects to a student-specific page (studentpage.html) upon success.

➤ Error Handling:

If the entered username or password is incorrect for either teachers or students, it displays an appropriate error message.

Plateforme professeur :

✓ Teacher's courses :

A screenshot of a web application's 'Add Course' modal. The modal has a title bar with 'Add Course' and a close button (X). It contains four input fields: 'Title:', 'Description:', 'Image URL:', and 'Price'. The 'Image URL:' field has a file selection button labeled 'Choisir un fichier' and a text label 'Aucun fichier choisi'. At the bottom right is a blue 'Submit' button.

here's a part from the JavaScript code which is responsible for managing courses on my webpage, including adding, deleting, and displaying them.

```
function updateCourseContainer(course) {  
    const id = course.id;  
    const titleValue = course.title  
    const description = course.description;  
    const priceValue = course.price;  
    const imgValue = course.image;|  
    const teacher = course.teacher;
```

The whole code function is:

- Upon the DOMContentLoaded event, it calls the function loadCoursesFromLocalStorage() to fetch courses from local storage.
- Upon the DOMContentLoaded event, it calls the function loadCoursesFromLocalStorage() to fetch courses from local storage.
- Defines functions openModal() and closeModal() to show and hide a modal dialog, respectively.
- Adds event listeners to the "Add Course" button and the close button within the modal to trigger modal opening and closing.
- Adds an event listener to the image input field (imageInput) to capture the selected image file.

- When an image is selected, it reads the file using FileReader and stores the base64-encoded image URL in the imageUrl variable.
- ❖ Form Submission:
 - Adds an event listener to the form (addCourseForm) submission.

When the form is submitted, it prevents the default form submission behavior.

- Retrieves form values such as title, description, and price.

Generates a unique ID for the new course.

- If an image is not selected, it assigns a default image URL.
- Creates a new course object with the collected data and adds it to local storage using addCourseToLocalStorage() function.
- Calls updateCourseContainer() to dynamically update the UI by adding the new course to the container.

Courses participants :

- each time a student registers for a course; he will be seen by the teacher of the course he has chosen with his name and slots he has chosen and his payment status

Plateforme étudiant :

✓ Courses for student to choose :



This enrollCourse() function handle the enrollment process for a specific course.

- The function is likely called when a user clicks on a button to enroll in a course. It retrieves the id attribute of the clicked button, which presumably represents the ID of the course to be enrolled in.
- Form Submission Handling:

- It adds an event listener to the enrollment form (enrollmentForm) submission.
- Upon form submission, it prevents the default form submission behavior.
- It retrieves values from the form fields such as motivation, paymentOption, and session. These values likely represent the user's motivation for taking the course, their preferred payment option, and session details.
- ✓ **Enroll in course :** Here's a breakdown of the script functionality to enroll for a course

DOMContentLoaded Event Listener:

- On page load, it calls the loadCoursesFromLocalStorage() function to fetch and display all courses stored in local storage.
- It sets up event listeners for modal functionality, payment option change, and enrollment.

Modal Functionality:

- It defines functions to open and close a modal dialog.
- Event listeners are set for closing the modal when the close button is clicked or when the user clicks anywhere outside the modal.

Payment Option Handling:

It sets up a function handlePaymentOptionChange() to show or hide the credit card field based on the selected payment option.

Updating Course Container:

The updateCourseContainer() function dynamically adds course information to the container.

It also checks if the current student is already enrolled in a course and changes the enrollment button accordingly.

Enrollment Functions:

- `enrollCourse(event)`:
 - Handles the enrollment process when the user clicks the enrollment button for a course.
 - It retrieves form values such as motivation, payment option, and session.
 - If the payment option is credit card, it retrieves additional card details.
 - It adds the enrollment information to local storage, closes the modal, reloads the courses, and refreshes the page.
- `deleteEnrollment(event)`: Handles quitting a course enrollment. Removes the enrollment from local storage, reloads the courses, and refreshes the page.

Local Storage Management:

Functions like `addEnrollmentToLocalStorage()` are responsible for adding enrollment information to local storage.

Payment Option:

Credit Card

Card Number:

Expiration Date:

CVV:

Cardholder's Name:

Envoyer

- It sets up a function `handlePaymentOptionChange()` to show or hide the credit card field based on the selected payment option.

✓ Edit student profile:

PROFILE

First Name

acila

Middle Name

laref

Academic Level

L3

Username

acila

Password

...

Save edit

The studentprofil.html code appears to be a form for editing a user's profile information anytime.

❖ Theme:

The theme.js code is responsible for managing the theme of the webpage.

UpdateTheme() Function:

- This function updates the theme of the webpage based on a condition (isDarkTheme).
- It retrieves the <link> element with the ID themeStylesheet, which represents the stylesheet responsible for the theme.
- Depending on the value of isDarkTheme, it updates the href attribute of the stylesheet to either a dark theme stylesheet or a light theme stylesheet.
- It also updates the icon displayed for the theme toggle button (themeIcon) based on the selected theme.

Theme Storage:

- After updating the theme, it saves the selected theme ('dark' or 'light') to the local storage using localStorage.setItem().


```
function updateTheme() {
  const stylesheet = document.getElementById('themeStylesheet');
  if (isDarkTheme) {
    stylesheet.href = 'styles/dark/styles.css';

    themeIcon.classList.remove('bi-moon');
    themeIcon.classList.add('bi-sun');
  } else {
    stylesheet.href = 'styles/light/styles.css';
    themeIcon.classList.remove('bi-sun');
    themeIcon.classList.add('bi-moon');
  }
  // Save theme to local storage
  localStorage.setItem('theme', isDarkTheme ? 'dark' : 'light');
}
document.getElementById('openPdfLink').addEventListener('click', function() {
  var pdfUrl = './Rapport source prof.etudiant.site.pdf';
  window.open(pdfUrl, '_blank');
});
```

White:

Learning +

PROFILE

HELP



Logout →

COURSES



1500 DA

poets

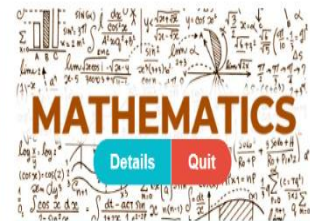
By nano



2100 DA

introduction to network

By sultana

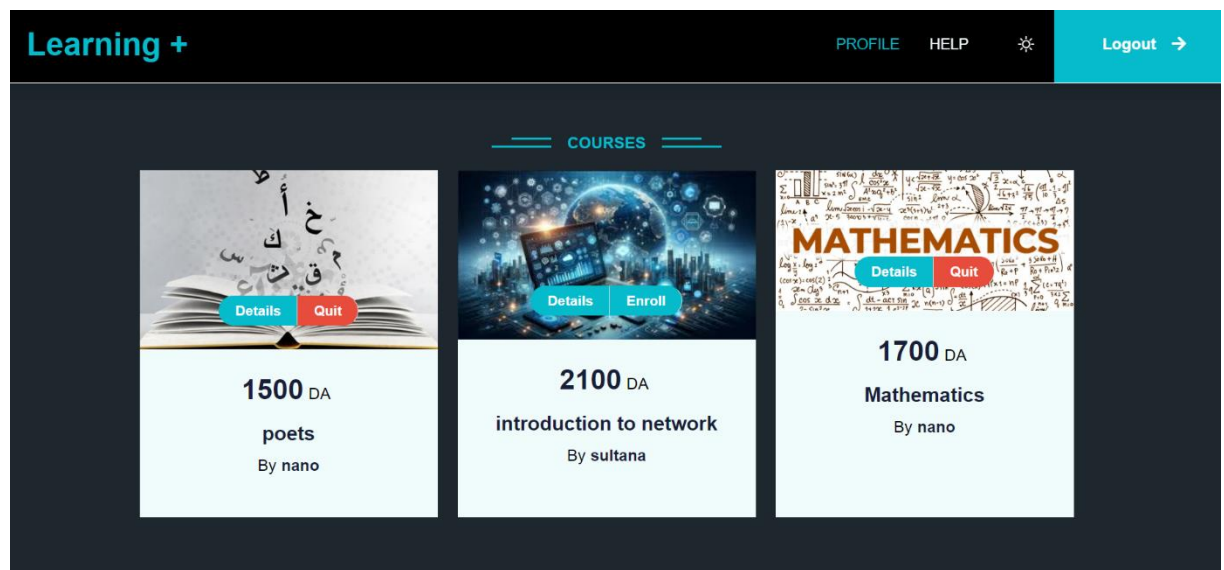


1700 DA

Mathematics

By nano

Black:



❖ Maintenance:

- To save and recover the data I used JSON local storage and manipulated the DOM. This clever approach allows me to store information persistently and interact with page elements dynamically.
- Using JSON local storage, I can easily convert the data into JSON format and save under a specific key. This allows me to store student information, such as their last name, first name, telephone number and payment status, in an organized manner.

Conclusion:

After the completion of our mini-project, I was able to achieve my goals of Satisfactory manner. Our platform has demonstrated its effectiveness and capacity to simplify student management.

Here are the results obtained:

- ✓ Time saving: Thanks to our innovative tool, I was able to automate many numerous administrative tasks.
- ✓ Improved organization: Our intuitive platform allowed me to maintain Optimal organization. Student data is stored in a manner Structured and accessible.
- ✓ Reduction of errors: Thanks to automated data entry and field validation, I was able to minimize human errors.